METHOD OF MAINTAINING LISTS OF NETWORK CHARACTERISTICS

Inventor: Livio Ricciulli

CROSS REFERENCE TO RELATED APPLICATION

This application claims the priority of U.S. Provisional Application No. 60/188,620 filed on March 13, 2000 which application is hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The field relates to detecting or documenting attacks on a network. More particularly, the field relates to detecting or documenting attacks such as flooding without imposing unreasonable demands on routers, such as demanding large amounts of state.

Description of Related Art

Two examples of widely used Transmission Control Protocol (TCP)-based protocols are hypertext transfer protocol (HTTP) and file transfer protocol (FTP). These two protocols are becoming more important for the exchange of information over the Internet and are affected by the "SYN flooding" type of denial-of-service attack. A denial-of-service attack on an Internet network by TCP "SYN flooding" hinders the signaling mechanism, called "handshaking," that is used to establish TCP connections. When such an attack occurs, the affected network resources, such as an Internet server, are degraded in their ability to handle message traffic, resulting in a denial-of-service condition.

A client computer and a server computer can establish a virtual connection using Transmission Control Protocol/Internet Protocol (TCP/IP) via handshaking, such as three-way handshaking.

The "SYN flood" attack takes advantage of the TCP/IP handshaking process by sending numerous SYN packets with false ("spoofed") return addresses to a communications port on a server. The server sends out a SYN-ACK message to each return address for each of these SYN packets. The SYN-ACK message is simply lost in the network. The server never receives any ACK messages back because there are no client systems at the spoofed return addresses. The server, therefore, keeps waiting in vain for an ACK message and may keep a queue entry allocated, for example, for several seconds. In sending out the SYN-ACK messages, the server uses up memory resources and queues a half-open connection for each spoofed SYN message. After a predetermined waiting period, the server times out waiting for a SYN message and closes the corresponding half-open connection. On many systems the time out values are on the order of approximately one second, so the server's connection request queue can be depleted relatively slowly. After the server has enough half-open connections to fill up its queue, the server will start to drop subsequent SYN messages, such that legitimate SYN connection requests start to be ignored. On certain systems, the allowable half-open connection queue space may be as little as eight connections.

Thus, SYN flooding attacks reduce (or eliminate) the ability of the targeted server system to respond to legitimate connection requests. An

attacker can generally leisurely fill the server's connection request queue before earlier SYN messages reach a time out condition. The SYN flooding denial-of-service attack, if not dealt with properly, requires very little computation and bandwidth commitment from malicious users. Although SYN flooding requires an attacker to continuously flood a target system (otherwise within a few minutes the target will revert to normal operation), it is difficult to trace to the source of the SYN packets. Thus, the SYN flooding technique remains a viable attack.

Potential loss of revenue caused by preempting reliable TCP communications is enormous, and therefore adequate mechanisms for dealing with SYN flooding are needed. Current SYN flooding defense mechanisms seem to have greatly mitigated the problem by making it harder for an attacker to negatively affect service. The most popular approach uses a "brute force" technique. In this approach, the TCP "connection pending" data structure (implementing the connection request queue) is made sufficiently large that an average attacker, to be successful, would need to flood connection requests at a rate exceeding reasonable bandwidth capabilities. This solution, although sometimes very practical, requires large amounts of protected kernel memory and may slow down the server response time for looking up connections in the vast "connection pending" data structure. Other less popular techniques use one-way hash functions (with Internet "cookies") to verify the authenticity of connection requests and therefore eliminate unnecessary memory allocation. Some of these latter techniques can introduce changes in the TCP signaling behavior and are

therefore less favored. Firewall approaches actively monitor the TCP signaling traffic to detect possible attacks and inject ad-hoc signaling messages in the network to mitigate the denial-of-service attack. These approaches are awkward because they introduce additional administrative complexity, may introduce significant delays for legitimate connection establishment, or may expose the system to different, though arguably less severe, kinds of vulnerabilities.

No one mechanism seems to provide an optimal solution, and thus a careful protection approach is usually constructed by using a combination of techniques. What is needed is a solution that can complement or replace existing solutions.

SUMMARY OF THE INVENTION

Various embodiments include a method of maintaining lists of network characteristics of messages. The messages can be detected. The messages can travel near at least the first network node. The messages can comprise network characteristics. The lists of network characteristics of the messages can be updated. The lists include instances of the network characteristics, based on a frequency of occurrences of the instances.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 shows one embodiment for maintaining one or more lists of one or more network characteristics.

FIG. 2 shows a flow chart of one embodiment for maintaining one or more lists of one or more network characteristics.

FIG. 3 shows a flow chart of an aspect of some embodiments for maintaining one or more lists of one or more network characteristics.

FIG. 4 shows a flow chart of an aspect of some embodiments for maintaining one or more lists of one or more network characteristics.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

Various embodiments can detect a source or sources of a flooding attack. One embodiment can identify such sources by sending a packet into the network which can follow the offending traffic upstream. The packet can follow the stream in the reverse direction by going to routers which have a record of the attacked destination address in their top-N most used destination addresses. Once the PROBE packet reaches the end of the path and, for any reason, cannot go any further, the last router will send report packet back to the destination.

Various embodiments of flooding detection can have one or more of the following features: (1) working at the IP level; (2) targeted to flooding in general; and/or (3) deployable without compromising anonymity. With some embodiments, real-time detection of the source of flooding attacks can be performed without requiring a large amount of state in routers.

Various embodiments have routers with one or more lists of top-N more seen or most seen network characteristics, for example, destination addresses, in a small cache. This list can vary with time relatively slowly, for

example, on the order of seconds for some embodiments. In some embodiments, the cache can have a number of instances of network characteristics substantially equal to or greater than C/F, where C can be a total aggregate capacity of a router, and F can be a minimum sustained flooding rate to detect. For example, to detect a 1MB/s flooding on a 1GB/s router, a cache of 1000 instances may by adequate. In one embodiment, listed instances can include a destination address and an ingress port.

When an attack, such as flooding, is detected, a message can be sent upstream by the attacked network node. For example, the message payload can contain a return address R, a network/host address H and/or a cookie generated by the attacked network node.

In some embodiments, an upstream router can look up H to check for a match in one or more lists of the local cache. If a match results, the router can forward a message upstream to appropriate interfaces. This can repeat recursively with routers further upstream.

In some embodiments, if an upstream router does not find H in the local cache, a report message can be sent to R with, for example, interface information of a downstream neighbor and the cookie. In some embodiments, this can be identified as an entry point of the attack, such as the flood.

If the upstream router does not implement this mechanism the router will send a report packet to R with its interface information and/or the cookie (this will be identified as the entry point of the flood).

R can receive the report message and/or cookie. This information can activate an upstream filtering defense. The relevant ISP or authorities can be notified.

An attacker can avoid being traced back by, for example, reducing the rate of packets and/or rotating destination addresses. The flooding can then be dispersed spatially and/or temporally, thus reducing bad effects of the attack. This can work for DDoS, since the message can be sent upstream on different interfaces in a multipath fashion. Filtering rules can be dynamically installed on an identified entry point, and/or the whole path. Some embodiments introduce hysteresis in the top-N list.

In another embodiment of this invention, the message can carry a cryptographic credential such as a certificate. When the message reaches a router which can send the report message back to the attacked network node, the credential can be verified. A filter can be set up to block traffic to destination H.

Figure 1 shows one embodiment for maintaining one or more lists of one or more network characteristics 100. A first network node 110 can be coupled to a first packet network 120 and a second packet network 130. Examples of packet networks include ATM and IP networks. The first network node 110, can be, for example, a router through which messages travel between the first packet network 120 and the second packet network 130. A second network node 140 can be coupled to the first network node 110. Other embodiments of the second network node 140 can be coupled differently, for example, coupled to one side of the first network node 110, or

coupled to more ports or routes coupled to the first network node 110. The second network node 140 can detect the messages traveling near the first network node 110. For example, the second network node 140 can detect messages traveling by or through the first network node 110. Some embodiments of the second network node 140 can record the messages traveling near the first network node 110. One embodiment of the second network node 140 includes a sniffer such as a packet sniffer to detect the messages traveling near the first network node 110. Some of the messages traveling near the first network node 110 can be, for example, recursive messages and/or multipath messages.

Figure 2 shows a flowchart 200 of one embodiment for maintaining one or more lists of one or more network characteristics. Various embodiments can alter, add to, delete from, and/or reorder elements of the flowchart 200. In 210, messages traveling a first packet network can be detected. The messages can have network characteristics. In 220, lists of network characteristics of messages can be updated, so that lists have instances of network characteristics based on frequency of occurrences of instances. The lists can have a group of more or most frequently occurring instances of the network characteristics. The frequency of occurrences can include a number of occurrences in an amount of time. In one embodiment, a number of instances in each of the lists can be substantially equal to or greater than a quotient. The quotient can include a capacity rate of a router (for example, first network node 110) divided by a threshold flooding rate.

There are many possible network characteristics that can be updated in 220. For example, IP source addresses 230, destination IP addresses 235, source TCP ports 240, source UDP ports 245, destination TCP ports 250, destination UDP ports 255, TCP flags 260, and/or ICMP flags 265.

Figure 3 shows a flowchart 300 of an aspect of some embodiments for maintaining one or more lists of one or more network characteristics. Various embodiments can alter, add to, delete from, and/or reorder elements of the flowchart 300. In 310, messages can be prevented from transiting the first network node. One embodiment prevents by filtering. Some embodiments prevent, responsive to receiving a message from an attacked network node. The attacked network node may have received a flooding attack and/or a denial of service attack. Such messages can have suspicious instances of network characteristics of lists. The suspicious instances can be associated with attacks on attacked network nodes. In 315, suspicious instances can be compared with repeatedly updated lists. If the compare fails to result in a match, prevention can be halted. One embodiment of halting the preventing can include removing the filter. In 320, lists can be repeatedly updated. Instances having low frequency of occurrences can be removed from lists. In various embodiments, the updating can occur at the second network node 140 an/or a third network node.

There are many possible network characteristics that can be matched in 3150. For example, IP source addresses 330, destination IP addresses

335, source TCP ports 340, source UDP ports 345, destination TCP ports 350, destination UDP ports 355, TCP flags 360, and/or ICMP flags 365.

Figure 4 shows a flowchart 400 of an aspect of some embodiments for maintaining one or more lists of one or more network characteristics. Various embodiments can alter, add to, delete from, and/or reorder elements of the flowchart 400. In 410, a message can be received. In some embodiments, the message can be received at the second network node 140 or a third network node. In some embodiments, the message can be received from an attacked network node or an intermediate network node. The message can include, for example, a cookie generated by the attacked network node. The message can be caused by an attacked network node. In 420, instances of network characteristics can be compared with the suspicious instance in the message caused by the attacked network node. If the compare results in a match 430, than 440 follows; else 450 follows. In 440, a message with the suspicious instance is forwarded towards suspect network nodes. The attack on the attacked network node possibly transited the suspect nodes. In some embodiments, the forwarded message can include updated information, for example, a network identifier of the forwarding node. In some embodiments, the forwarding can be based at least on an instance of a network characteristic maintained in a list. In 450, a report message can be sent towards the attacked network node.

Various embodiments can comprise elements coupled together physically and/or functionally. The various embodiments of the structures and methods that are described above are illustrative only and do not limit

the scope to the particular embodiments described. For example, in view of this disclosure, those skilled in the art can define other variable names, data structures, protocols, etc., and/or other software and/or hardware to use these alternative features, creating a method or apparatus such as a software and/or hardware apparatus. The following claims cover such alternatives.